MAPUA INSTITUTE OF TECHNOLOGY

SCHOOL OF EE-ECE-CpE

Gate-Level Modeling

DRILL 3

NAME:

STUDENT NUMBER:

TERMINAL NUMBER:

DATE OF PERFORMANCE:

DATE OF SUBMISSION:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

PROFESSOR

I. DISCUSSION

Gate-level or structural modelling describes a circuit by specifying its gates and how they are connected with each other. It provides a textual description of a schematic diagram. Verilog HDL includes 12 basic gates as predefined primitives. Four of these primitive gates are of the three-state types. The other eight are all declared with the lowercase keywords **and**, **nand**, **or**, **nor**, **xor**, **xnor**, **not**, and **buf**. N-input primitives such as **nand** and **or** can have any number of scalar inputs (e.g., a three-input and gate) while n-output primitives such as **not** and **buf** can drive multiple output lines from a single input distinguished by their identifiers.

The following are the truth tables for the predefined primitive gates **and**, **or**, **xor**, and **not**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| and | 0 | 1 | x | z |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | x | x |
| x | 0 | x | x | x |
| z | 0 | x | x | x |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| or | 0 | 1 | x | z |
| 0 | 0 | 1 | x | x |
| 1 | 1 | 1 | 1 | 1 |
| x | x | 1 | x | x |
| z | x | 1 | x | x |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| xor | 0 | 1 | x | z |
| 0 | 0 | 1 | x | x |
| 1 | 1 | 0 | x | x |
| x | x | x | x | x |
| z | x | x | x | x |

|  |  |  |
| --- | --- | --- |
| not | input | output |
|  | 0 | 1 |
|  | 1 | 0 |
|  | x | x |
|  | z | x |

Remember that the output is always listed first in the port list of a primitive, followed by the inputs.

One instance of the structural modelling under Verilog HDL is the first example from the first drill (Drill1\_1.vl). More of the examples are given below.

II. Drill Exercises

A. Draw the logic diagram of the following Boolean function, and simulate using Icarus Verilog HDL.

F (V, W, X, Y, Z) = ∑ (0, 2, 4, 6, 9, 13, 21, 23, 25, 29, 31)

After simplifying the function using a five-variable K-map, we obtain

F = V’W’Z’ + WY’Z + VXZ



Fig 2.1

*//Verilog model of circuit of Fig 2.1*

*module circuit2\_1(V, W, X, Y, Z, out1);*

*input V, W, X, Y, Z;*

*output out1;*

*wire Vnot, Wnot, Xnot, Ynot, Znot, or1, or2, or3;*

*not U1(Vnot, V), U2(Wnot, W), U3(Xnot, X);*

*not U4(Ynot, Y), U5(Znot, Z);*

*and U7(or1, Vnot, Wnot, Znot), U8(or2, W, Ynot, Z);*

*and U9(or3, V, X, Z);*

*or U6(out1, or1, or2, or3);*

*endmodule*

*module test2\_1;*

*wire w1;*

*reg x1, x2, x3, x4, x5;*

*circuit2\_1 test2\_1(x1, x2, x3, x4, x5, w1);*

*initial begin*

*x1=1'b0; x2=1'b0; x3=1'b0; x4=1'b0; x5=1'b0; //0*

*#100 $finish;*

*end*

*initial begin*

*#2 $display(x1,x2,x3,x4,x5," ",w1);*

*#2 x1=1'b0; x2=1'b0; x3=1'b0; x4=1'b1; x5=1'b0; //2*

*#2 $display(x1,x2,x3,x4,x5," ",w1);*

*#2 x1=1'b0; x2=1'b0; x3=1'b1; x4=1'b0; x5=1'b0; //4*

*#2 $display(x1,x2,x3,x4,x5," ",w1);*

*#2 x1=1'b0; x2=1'b0; x3=1'b1; x4=1'b1; x5=1'b0; //6*

*#2 $display(x1,x2,x3,x4,x5," ",w1);*

*#2 x1=1'b0; x2=1'b1; x3=1'b0; x4=1'b0; x5=1'b1; //9*

*#2 $display(x1,x2,x3,x4,x5," ",w1);*

*#2 x1=1'b0; x2=1'b1; x3=1'b1; x4=1'b0; x5=1'b1; //13*

*#2 $display(x1,x2,x3,x4,x5," ",w1);*

*#2 x1=1'b1; x2=1'b0; x3=1'b1; x4=1'b0; x5=1'b1; //21*

*#2 $display(x1,x2,x3,x4,x5," ",w1);*

*end*

*endmodule*

B. Simulate the function of a 4x1 line multiplexer using primitive logic gates.

Before solving this problem, you need to recall first the logic diagram of a four-to-one-line MUX.



*module test2\_2;*

*reg [3:0]A;*

*reg [1:0]sel;*

*wire Out2;*

*initial begin*

*$display(" TIME A select Output");*

*$monitor($time,,,,,,,"%h %d %b",A, sel, Out2);*

*#50 A=4'hE; sel=2'b00;*

*#50 A=4'hA; sel=2'b11;*

*#50 A=4'hC; sel=2'b10;*

*#50 A=4'hB; sel=2'b01;*

*#50 A=4'h0; sel=2'b01;*

*#50 $finish;*

*end*

*circuit2\_2 t2\_2(A, sel, Out2);*

*endmodule*

*module circuit2\_2(input [3:0]I, input [1:0]S, output Out1);*

*wire [3:0]In;*

*wire S0not, S1not;*

*not #(2) (S0not, S[0]);*

*not #(2) (S1not, S[1]);*

*and #(5) (In[0], I[0], S0not, S1not);*

*and #(5) (In[1], I[1], S[0], S1not);*

*and #(5) (In[2], I[2], S0not, S[1]);*

*and #(5) (In[3], I[3], S[0], S[1]);*

*or #(5) (Out1, In[0], In[1], In[2], In[3]);*

*endmodule*

III. Programming Exercise

1. Display the truth tables of **buf**, **nand**, and **xnor** in your prompt by creating the necessary Verilog HDL program. Save as exercise3\_1.vl file.
2. Develop and simulate a gate-level model of a circuit that generates the 9’s complement of a BCD digit. Show the outputs of all possible input combinations. Save as exercise3\_2.vl file.
3. Construct a two-bit by two-bit binary multiplier by using half-adders/full-adders. Design your half-adder using gates, and then simulate the circuit using Verilog HDL. Save as exercise3\_3.vl file.

IV. Review Questions

1. Complete the following truth tables below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| xnor | 0 | 1 | x | z |
| 0 |  |  |  |  |
| 1 |  |  |  |  |
| x |  |  |  |  |
| z |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| nand | 0 | 1 | x | z |
| 0 |  |  |  |  |
| 1 |  |  |  |  |
| x |  |  |  |  |
| z |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| nor | 0 | 1 | x | z |
| 0 |  |  |  |  |
| 1 |  |  |  |  |
| x |  |  |  |  |
| z |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| buf | input | output |
|  | 0 |  |
|  | 1 |  |
|  | x |  |
|  | z |  |

1. How do gate delays affect the simulated output of a gate model program?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. What are the advantages and disadvantages of using gate-level models?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_